ASCoT, the NASA Analogy Software Cost Tool Suite: Expanding Our Estimation Horizons

Jet Propulsion Laboratory, California Institute of Technology
Pasadena, CA 91109

Jairus Hihn, Tom Youmans, Alex Lumnah, Michael Saing, Elinor Huntington, Melissa Hooke
Jet Propulsion Laboratory/ California Institute of Technology
Pasadena, CA

James Johnson
National Aeronautics and Space Administration
Washington, DC

Tim Menzies
North Carolina State University
Raleigh, NC

***Abstract*—The NASA Analogy Software Costing Tool Suite (ASCoT) consists of a cluster-based analogy estimator for estimating software development effort, a K-Nearest Neighbors (KNN) analogy estimator for estimating effort and delivered lines of code, a simple regression-based cost estimating relationship (CER) model that estimates cost in dollars, and a probabilistic version of COCOMO II. In this paper we document the analogy algorithms as well as summarize the results of the performance of the KNN and the principle components (PCA) cluster analogy models. KNN performance is assessed by varying the number of inputs and number of neighbors. Four different clustering methods: K-means, Spectral Clustering, Hierarchical Clustering, and Principle Components Analysis (PCA), and their respective evaluation criterion are described in detail. The comparative performance of all four estimation models is assessed using magnitude of relative error (MRE) measurements.**

TABLE OF CONTENTS

1. INTRODUCTION

Software cost estimates are often required in the early stages of mission design when the technical details are not fully understood and software experts are not around. To help address this problem, a decade-long research journey was undertaken to better understand alternative estimation methods and to determine if a model could be developed with minimal, flexible, inputs while avoiding the occurrence of large estimation errors, even when systems engineers and non-experts are attempting to estimate flight software costs.

The use of analogy has been well documented in the literature on software cost estimation [1,2,3]. Software developers typically perform analogy estimates based purely on memory with no supporting data, which results in non-reproducible estimates with difficult to determine accuracy. Thus, academics have proposed various formalizations of analogy to improve estimation replicability and accuracy [2,3, 8]. The simplest of these formulations use distance as a means of assessing similarity between projects, frequently using data mining algorithms such as K-Nearest Neighbors or clustering. An advantage of these methods is their ability to handle categorical data in addition to quantitative data.

The NASA Analogy Software Cost Tool (ASCoT)[1], was first introduced at the 2016 IEEE Aerospace conference [4] and as a web-based tool [5] in 2017. ASCoT is designed to address the specific problems associated with generating more realistic estimates in the earliest parts of the lifecycle (CML 1). The NASA Analogy Software Cost Model is built on research into the effectiveness of data mining algorithms to develop repeatable, well-documented analogical software estimation models [6,7,8,9]. The purpose of ASCoT is to enable the ability to estimate software development effort and cost early in the project lifecycle using easily attainable inputs, such as the type of mission and the number of instruments. ASCoT is developed as a compliment or extension to the existing widely applied parametric methods. The other contribution of this paper is the emphasis on the use of the magnitude of relative error (MRE) as a metric for evaluating cost model performance for cluster analysis, and for comparing parametric vs non-parametric models[2].

[1] For full list of acronyms used in this paper, see Appendix A.
[2] Pred(30) is an example of a MRE metric that was a popular measure of model performance in the eighties and nineties, but seems to have since fallen out of favor.

The following is a summary of the key findings from our previous work [4,5,6,7,8,9]:

- There are a variety of models whose performances are hard to distinguish (given currently available data), but some models are clearly better than others.

- Based on an extensive analysis of various estimation models, it was found that COCOMO II performed as well or better than every other estimation method evaluated, which included various data mining algorithms [10,11,12]. In other words, if one has sufficient detailed knowledge to run COCOMO or a comparable parametric model, then the best model is the parametric model.

- When insufficient information exists then a model using system parameters can be used to estimate software costs with only a small reduction in accuracy. The main weakness is the possibility of occasional large estimation errors, which the parametric model does not exhibit.

- While a nearest neighbor model performs as well as clustering based on MMRE, clustering handles outliers better and provides a structured model with more capability.

ASCoT the tool suite consists of four estimation models/tools: an Analogy effort estimator based on Clustering, an Analogy effort and size estimators based on KNN, a linear regression CER (Cost Estimating Relationship), and a probabilistic version of COCOMO II. However, ASCoT is only available via the NASA ONCE server which is only assessable to those with a NASA Badge. Therefore, the focus of this paper is primarily on documenting the algorithms and validation results for the PCA cluster analysis and the KNN estimation models which can be applied by others.

2. DATA SUMMARY

Data Sources

The primary data source is the NASA Cost Analysis Data Requirement (CADRe). The CADRe is a formal project document that describes the life-cycle cost, schedule, technical, and risk information of a project. The CADRe has three separate Parts: A, B, and C. Part A is a narrative description of the project throughout its lifecycle at each milestone and includes essential subsystem descriptions, bock diagrams, and heritage assumptions. Part B contains the technical design parameters such as power, mass, and software metrics for each subsystem in a standardized template. Part C captures all the cost data broken out by Level 2 of the Work Breakdown Structure (WBS) throughout the lifecycle by project phase. Questionable CADRe data was revised with information/data from other sources and additional data added as follows:

- Available missing data items were obtained from other sources including contacting project software managers
- System descriptor data was supplemented with data from NASA project websites, project reports, and Wikipedia articles.
- Software metrics for older missions that predated the CADRe were supplemented with data records from a data collection conducted for the International Space Station that was completed in 1990. A subset of these records can be found at the PROMISE (Predictor Models in Software Engineering) website under the COCOMO directory.
- Contributed NASA Center level data

Data Description

A list of relevant variables used in the tool can be found in Table 1. Each variable is accompanied by the number of missions with complete data for that field.

Table 1 Data summary with number of records – 34 missions have complete verified data and are used in ASCoT Clusters

Data Item	Number of Data Records as of 2018
Total development effort in work months	39
Flight Software Development Cost	43
Flight System Development Cost	43
Source Lines of Code (SLOC)	
Delivered SLOC	51
Inherited SLOC (Reused plus Modified reused)	43
COCOMO Model inputs (See Appendix A for the parameter definitions) - Translated from CADRe	19
Systems Parameters	
Mission Destination (Asteroid/Comets, Earth, Inner (planetary), Outer (planetary))	51
Multiple element (probe, etc…)	51
Number of Instruments	51
Number of Deployable	51
Flight Computer Redundancy (Dual Warm, Dual Cold, Single String)	51
Software Reuse (Low, Medium, High)	49
Software Size (Small, Medium, Large, Very Large)	51

The data used in the ASCoT estimation tools was last updated in March, 2018. Note that the number of records reported in the data summaries in Table 1 through Table 14 vary due to missing data.

There are a total of 61 missions in the dataset, with 51 that could be used in at least one model. While the models share many of the same missions, the data used is different as ASCoT contains different models that estimate effort, dollars, and lines of code.

As few new missions launch each year, the number of records has only increased by nine since the ASCoT prototype was developed in 2015. Since then, the focus has been on improving data quality, improving model performance and adding new estimation models. For a detailed description of the types of data parameters collected see Appendix B and for the COCOMO model see [13]. Appendix C contains a list of all missions for which data was obtained with an indication of which missions were used to build the analogy, KNN and regression models.

Table 2 through Table 7 below summarize the data by median, average, and spread metrics for each parameter. There has been little change in the summary metrics as a result of the addition of the new and corrected data. Overall, inner/outer planetary missions have more lines of code, higher development effort, cost more, have more instruments, and are more likely to be dual string than Earth Orbiters. Not surprisingly, inner/outer planetary missions have significantly more deployables and instruments than all other mission types. Slightly surprising is that Earth Orbiters and inner/outer planetary missions have similar inheritance rates even though many Earth orbiters can draw more easily on the various contractor product lines.

Table 2 Effort by Mission Type

Mission Type	EFFORT (Months)				
	# of Records	*Median*	*Std. Dev.*	*Avg.*	*Range*
Asteroids / Comets	7	546	373	583	48 - 1048
Earth	14	499	466	632	100 - 1830
Inner Planetary	17	664	435	813	336 - 1888
Outer Planetary	4	620	411	723	346 - 1307

Table 3 Delivered SLOC by Mission Type

Mission Type	Delivered SLOC by mission type, actual count				
	# of Records	*Median*	*Std. Dev.*	*Average*	*Range*
Asteroids/ Comets	7	143,000	35,189	118,679	24,100 – 246,654
Earth	23	62,000	39,986	211,600	23,000 – 170,000
Inner Planetary	17	122,000	133,765	105,411	62,900 – 475,000
Outer Planetary	4	54,000	21,633	126,120	24,000 – 130,150

Table 4 Software size by size category and mission type

Mission Type	Software Size Category					
	# of Records	*Small*	*Medium*	*Large*	*Very Large/ Extra Large*	*Median*
Asteroids/ Comets	7	2	2	2	1	Medium
Earth	23	2	16	5	0	Medium
Inner Planetary	17	0	5	9	3	Large
Outer Planetary	4	1	1	2	0	Large

Tables 4 and 5 show software size and inheritance by mission type. While the actual code counts for software size and inheritance (or at least estimated code percentage for inheritance) were known data parameters, these values were binned into categories for two reasons. Most notably, the tool is designed to be used for early lifecycle phase estimates only, so estimators would only require an approximate idea about the number of delivered and inherited SLOC. The other reason is due many inconsistencies in how lines of code are recorded in the NASA CADRe and with a lack of documentation on counting rules, the use of categories is a more accurate reflection of the actual accuracy of the data.

Table 5 Inheritance by Mission Type

Mission Type	Inheritance						
	# of Records	*VL to None*	*Low*	*Med-ium*	*High*	*Very High*	*Median*
Asteroids / Comets	7	1	1	3	0	2	Med-ium
Earth	21	3	1	5	6	6	High
Inner Planetary	17	4	1	3	4	5	Med-ium
Outer Planetary	4	2	0	1	1	0	Med-ium

Table 6 and 7 show deployables, instruments, and flight computer redundancy by mission type. It is shown that the number of deployables and number of instruments are higher for inner/outer planetary compared to Earth orbiters and asteroids/comets. In addition, Dual-String Cold and Dual String Warm flight computer redundancy is also higher compared to Earth missions.

Table 6 Deployables and Instruments by Mission Type

Mission Type	*# of Records*	Deployable		Instruments	
		Median	*Range*	*Median*	*Range*
Asteroids/ Comets	7	1	0 - 3	3	2 - 5
Earth	23	2	0 - 8	3	1 - 10
Inner Planetary	17	2	0 - 10	4	3 - 10
Outer Planetary	4	3	0 - 8	10	7 - 12

Table 7 Flight computer redundancy by mission type

Mission Type	# of Records	Flight Computer Redundancy			
		Single String	*Dual String-Cold*	*Dual String-Warm*	*Median*
Asteroids/ Comets	7	1	6	0	Dual String Cold
Earth	23	12	11	0	Single String
Inner Planetary	17	5	8	4	Dual String Cold
Outer Planetary	4	0	2	2	Dual String Warm

Tables 8 through 11 show Delivered Productivity by Logical Lines of Code by mission type and inheritance level; low (<20%), medium (<50%), high to very high (>=50%). Inherited code includes both reused and modified reused code reuse. As expected, all mission categories clearly show that increases in inheritance result in higher productivity rates.

Table 8 Productivity (Delivered Logical SLOC) by mission type

Mission Type	# of Records	Logical Equivalent SLOC			
		Median	Std. Dev.	Avg.	*Range*
Asteroids/ Comets	7	175	203	267	124 - 615
Earth	14	192	208	250	46 - 823
Inner Planetary	16	244	90	237	65 - 394
Outer Planetary	4	178	109	174	37 - 302

Table 9 Very low to none and low inheritance delivered productivity

Mission Type	Very Low to None and Low Inheritance (0% - <20%) Delivered SLOC Productivity			
	# of Records	*Avg. Prod*	*Median Prod*	*Range*
Asteroids/ Comets	2	89	89	24.1 - 154
Earth	4	112	117	62 - 150
Inner Planetary	5	214	149	62.9 - 475
Outer Planetary	2	77	77	24 - 130.15

Table 10 Medium inheritance delivered productivity

Mission Type	# of Records	Medium Inheritance (>20% - <50%) Delivered SLOC Productivity		
		Avg. Prod	Median Prod	Range
Asteroids/ Comets	2	141	141	100 - 182
Earth	3	91	81	23 - 170
Inner Planetary	1	-	-	-
Outer Planetary	1	-	-	-

Table 11 High and very inheritance delivered productivity

Mission Type	# of Records	High and Very High Inheritance (>=50%) Delivered Productivity		
		Avg. Prod	Median Prod	Range
Asteroids/ Comets	2	166	166	86 - 247
Earth	12	93	92	41 – 154
Inner Planetary	10	157	163	90 - 224
Outer Planetary	1	-	-	-

12 and 13 provide a summary of the flight software and flight system cost records in FY16 dollars ($M). As with the effort data, the cost of inner/outer planetary missions are more expensive then earth orbiters. The data indicates that the difference in cost is greater than the difference in effort between mission types. This is most likely because the reported cost includes procurements and costs of additional WBS elements that are not included in the effort data. For example, some contractors include simulators for the flight system with flight software costs as they are used for testing the flight software. Another pattern not shown here, but present in the data, is that the median value of the ratio of flight software costs to flight system cost is 10% for all mission types except In Situ which is 5%.

Table 12 Software development cost (FY16$ M)

Mission Type	*# of Records*	Software Development Cost (FY16$M)		
		Avg. Prod	*Median Prod*	*Range*
Asteroids/ Comets	6	12	12	1 - 24
Earth	19	9	7	1 - 24
Inner Planetary	16	19	17	3 - 63
Outer Planetary*	2	26	26	12 - 40

Table 13 Total spacecraft (FY16$ M)

Mission Type	*# of Records*	Total Spacecraft Development Cost (FY16$M)		
		Avg.	*Median*	*Range*
Asteroids/ Comets	6	173	166	50 - 305
Earth	19	123	72	15 - 386
Inner Planetary	16	287	217	41 - 1,335
Outer Planetary	2	297	297	193 - 401

3. ARCHITECTURE AND METHODOLOGY

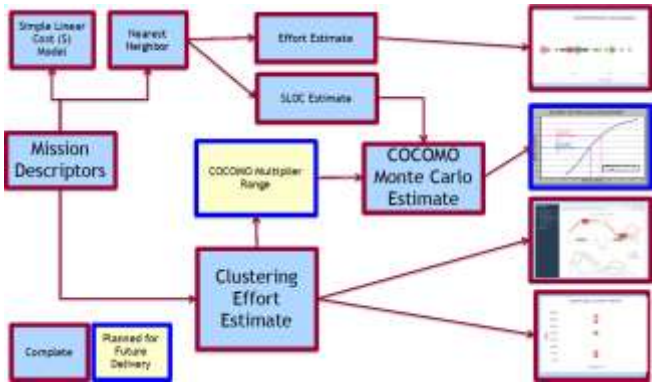

Figure 1 ASCoT Architecture

As mentioned previously, ASCoT consists of four estimation models/tools: a clustering model, a KNN model, a linear regression, and a probabilistic version of COCOMO II. How these tools relate to one another is shown in the architecture diagram in Figure 1. Except for COCOMO II, all of the models/tools in ASCoT only use high level system descriptors as inputs. When ASCoT is completed, the outputs of these models will be used as the inputs into the more complex COCOMO estimation model.

One of the significant contributions of the research conducted in developing ASCoT is the recognition of the importance of using the magnitude of relative error (MRE) and its associated measures, such as the mean or median MRE, as a metric for evaluating cost model performance across very different types of models. MRE measures are popular in the data mining literature because they require no assumptions about the underlying distributions, enabling one to compare model performance across very different types of models. Pred(30), another MRE statistic, was very popular in the cost field in the eighties and nineties but seems to have fallen out of favor [7]. For a detailed description of the various models evaluated and the MRE evaluation method see [2,4,7]. In generating the MRE and other evaluation metrics for this paper, we have used leave-one-out cross validation and then computed the average or median for each test point as appropriate. Detailed descriptions of procedure and analyses of model performance for each of the estimation tools is described in the sections below.

4. ASCOT ESTIMATION TOOLS

A. Cost Estimating Relationship

The following regression models were developed with cost data from the CADRe Part C with minimal normalization, partly as a test to see to what extent the raw CADRe data could be successfully used to develop a basic cost model. Minimal normalization makes verifying the project data used in the model very fast and easy. The basic regression models performed so well that they were included as part of the ASCoT tool. Again, as with the analogy model, the guiding principle was to keep the regression simple with inputs that can be "approximated" in the early stages of concept development through Step 1 proposals, and in early phases of the lifecycle. ASCoT features one univariate CER model which branches into three separate regressions; and one multivariate linear model with just two predictors.

Model 1: Software Development Cost as a function of Total Spacecraft Development Cost

Although this model is only dependent on one variable (Total Spacecraft Development Cost), it is actually the composition of three independent models segregated by mission type (All Missions, Inner and Outer Planetary Missions, Earth Missions). Each model stands independently, using only its subset of observations to fit a least squares simple linear regression. The model parameters and key test statistics for all three models are shown in Table 14.

All three models produced significant F-test results compared to the intercept-only model. As a quick rule of thumb, the results indicate that flight software costs around $4 million at a minimum and then runs at 5% of spacecraft cost. This result is heavily driven by the planetary missions, as when analyzing only Earth/ Lunar orbiter missions, the intercept is not significantly different from 0 and software runs at 6% of spacecraft cost.

Table 14 Univariate model parameters and test statistics

	Intercept (t)	β Spacecraft development cost (t)	Model statistics
All Missions	4.42 (3.34*)	0.05 (10.19*)	n = 42 F = 103.9* df = (1,41) R^2_{adj} = 0.71
Planetary Missions	7.24 (2.89*)	0.04 (6.77*)	n = 17 F = 45.9* df = (1,16) R^2_{adj} = 0.73
Earth Missions	1.36 (0.90)	0.06 (6.29*)	n = 18 F = 39.5* df = (1,17) R^2_{adj} = 0.68

Model 2: Software Development Cost as a function of Total Spacecraft Cost and Total Number of Instruments

In the multivariate case, all missions' Software Development Costs were regressed on both Total Spacecraft Development Cost and Total Number of Instruments using least squares minimization. However, as shown in Table 15, the result was not significant compared to the univariate model according to the a nested F-test which returned an F-statistic of 1.2, indicating that the addition of number of instruments to the model did not explain a significant amount of additional variance.

Table 15 Multivariate model parameters and test statistics

Intercept (t)	β Spacecraft development cost (t)	β Number of Instruments (t)	Model statistics
2.80 (1.40)	0.04 (7.85*)	0.53 (1.09)	n = 42 F = 103.9* df = (1,41) R^2_{adj} = 0.71 F_{nested} = 1.19

Outlier Analysis

Visualizations of the data for this model made obvious that the data included one clear outlier: Mars Science Laboratory (MSL). MSL is a very large Rover Mission whose Total Spacecraft Cost of $1,335 million and a Software Development Cost of $63 million make it an outlier in both the x and y directions. This value is extremely influential according to Cook's Distance, crossing the typical threshold value of 1, while the rest of the points average a Cook's Distance of .016. MSL also had high leverage, with a Hat Value of .70 which is well over the threshold of .095 (the typical threshold for Hat Values is 2p/n, where p is the number of predictors in the model).

An analysis of the relative error distributions of each model was performed to determine whether including the MSL outlier was justified. For the All Missions univariate model, while the median relative error for the curve with MSL was actually slightly lower than the MRE without MSL, it was only by a factor of .005 (.323 vs. .328), which is insignificant. Also, the MRE curves shown in Figure 2 are almost on top of each other, crossing over each other frequently, indicating that the regression models with and without MSL are indistinguishable in terms of the prediction errors that they produce.

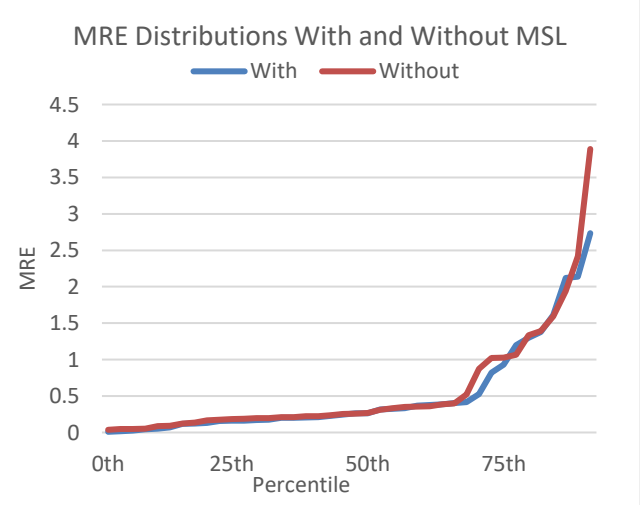

Figure 2 MRE distributions with and without the outlier

As the difference in error was negligible, the benefit of being able to predict MSL was worth including the outlier so that future predictions in the range of MSL are not drastically overestimated. The same results were found for the multivariate model and the Planetary Mission regression models, so MSL was included in the data.

B. Analogy Estimation Using KNN

Introduction—The ASCoT k-Nearest Neighbor regression model is a simple non-parametric method used to estimate either the total development effort or the total delivered lines of code using Euclidean distance as a measure of similarity.

A user may input any combination of the following variables to make an estimate: Mission Size, Mission Type, Redundancy, Destination, Number of Instruments, Number of Deployables, and Inheritance (if estimating total development effort). The decision to use these organization-specific variables comes from the authors of [3] who suggest that when using analogy methods on an intraorganizational database the specificity of the data is not as beneficial if the organization uses generic COCOMO-like factors. By using these organization-specific measurements available in the early phases of project formulation, we hope that the analogy methods used in ASCoT will not only provide accurate cost estimates, but also return relevant data from historical missions that are actually similar to the project at hand.

The algorithm takes in all of the provided dimensions and first converts the categorical variables to integers, per Table 22. The training data, described in the Data Summary section above, as well as the test point (the point that is being estimated) are then normalized by subtracting the mean of each dimension and dividing by the range of that dimension in the training data. This standardization method is similar to methods used in [2,3]

KNN Algorithm

The K-Nearest Neighbor (KNN) algorithm is a simple non-parametric model used to estimate a continuous dependent variable for a point based on the distance to neighboring points in the set [14].

Take a finite set $A \subset \mathbb{R}^{m+1}$, where $|A| = n$ for some $n \in \mathbb{Z}^+$ and each vector $\boldsymbol{v_i} \in A$ is of the form:

$$\boldsymbol{v_i} = (X_{i,1}, X_{i,2}, \cdots, X_{i,m}, Y_i) \quad (1)$$

where Y_i is the continuous dependent variable we are trying to estimate. Given a test point $\boldsymbol{p} \in \mathbb{R}^m$ of the form:

$$\boldsymbol{p} = (X_{p,1}, X_{p,2}, \cdots, X_{p,m}) \quad (2)$$

We define the distance D from the test point $\boldsymbol{p}$ to any training point $\boldsymbol{v_i}$ using the following function:

$$D(\boldsymbol{p}, \boldsymbol{v_i}) = \sqrt{\sum_{j=1}^{m}(X_{p,j} - X_{i,j})^2} \quad (3)$$

By calculating the distance D between the test point $\boldsymbol{p}$ and all of the neighbors $\boldsymbol{v}_i \in A$, we may reorder the set A such that:

$$D(\boldsymbol{p}, \boldsymbol{v_1}) \leq D(\boldsymbol{p}, \boldsymbol{v_2})) \leq \cdots \leq D(\boldsymbol{p}, \boldsymbol{v_n}) \quad (4)$$

Given some $k \in \mathbb{Z}^+$ such that $0 < k < n$, we may estimate the dependent variable Y_p for our test point according to the following equation:

$$Y_p = \frac{\sum_{i=1}^{k} \frac{Y_i}{D(\boldsymbol{p}, \boldsymbol{v_i})}}{\sum_{i=1}^{k} \frac{1}{D(\boldsymbol{p}, \boldsymbol{v_i})}} \quad (5)$$

The equation above provides an average of the dependent variable Y_i belonging to the k points closest to $\boldsymbol{p}$, weighted by the inverse distance to each point. Weighting by the inverse distance gives the points which are closest to the test point $\boldsymbol{p}$ the most influence over the estimate of $\boldsymbol{p}$.

Choosing the Number of Neighbors (k)

Leave-one-out cross validation (LOOCV) was utilized in order to choose an appropriate k value for which the prediction errors — both the magnitude of relative error (MRE) and the squared error (SE) — were minimized. LOOCV for selecting k according to this criterion operates according to the following algorithm:

1. Set *k*
2. Remove one observation, the "test mission", from the data
3. Build KNN model on the remaining observations or "training missions"
4. Predict value for the test mission using weighted average of *k* neighbors
5. Compute relative error and squared error for the test mission
6. Repeat Steps 2-5 for all *n* missions
7. Compute MRE and MSE summary statistics given *n* errors in Step 5 and 6
8. Repeat 1-7 for each new k and choose the k that minimizes the most parameters in Step 7

This form of cross validation is preferred over k-fold cross validation for the data at hand because it maximizes the number of observations available to train the KNN algorithm. That said, LOOCV still prevents the test mission from influencing its own cost estimation, which means that these error rates are very similar to the error rates that users should expect when predicting mission costs for new missions that are not in the dataset. Table 16 and Table 17 summarize the MRE and MSE distributions obtained from LOOCV for k = 1, 2, …, 7 and for predicting using the mean.

Generally, as k is incremented, each of the MRE summary statistics decreases gradually, hits a minimum, then increases. Since the goal is to maximize the number of missions with small MREs, we want the distribution of the MREs to be as close to zero as possible. The same is true of the squared error statistics, MdSE and MSE.

Table 16 KNN Effort k value LOOCV error rates

	MRE					SE	
k	25th	50th	Mean	75th	Max	Median	Mean
1	0.18	0.47	0.60	0.60	3.93	67600	128415
2	0.18	0.38	0.54	0.54	4.22	64803	100932
3	0.15	0.37	0.51	0.51	4.50	39814	92299
4	**0.13**	**0.31**	**0.47**	**0.47**	**3.67**	**37351**	**88276**
5	0.19	0.31	0.51	0.51	3.53	36727	102908
6	0.16	0.31	0.52	0.52	3.84	36601	103112
7	0.15	0.28	0.52	0.52	3.85	40621	106915
Mean	0.17	0.37	0.69	0.69	6.03	40161	154793

For the KNN Effort model, four of the seven LOOCV error statistics reached a minimum at k=4: the 25th and 75th percentiles of the MRE, the mean MRE, and the MSE (mean squared error). The three error statistics that were not minimized fell to within 10% of their minimum values. The median MRE for k=4 was .31, indicating that half of test missions were estimated within 31% of their actual efforts, while half of test missions fell within 193 work months of their actual effort according to MdSE.

Table 17 KNN SLOC k value LOOCV error rates

	MRE					SE	
k	25th	50th	Mean	75th	Max	Median	Mean
1	0.14	0.35	0.60	0.60	4.47	1479	4020
2	**0.21**	**0.34**	**0.61**	**0.61**	**4.12**	**1159**	**3741**
3	0.21	0.31	0.63	0.63	5.32	1805	3788
4	0.21	0.31	0.59	0.59	5.03	1560	3958
5	0.15	0.32	0.59	0.59	5.51	1852	4132
6	0.17	0.32	0.61	0.61	5.78	1774	4546
7	0.14	0.33	0.63	0.63	5.48	1924	4823
Mean	0.21	0.35	0.70	0.70	4.53	1375	6143

For the KNN SLOC model, the evaluation criterion did not point as clearly to one particular value for k. Since the MRE values did not exhibit the same trend toward a minimum for SLOC as they did for the Effort model, we relied on the squared error values to choose k.

Both the MdSE and the MSE reached a minimum at k=2. The MdSE for k=2 was 1159 (thousand SLOC), which was 22% lower than the next lowest k value. In addition, the maximum MRE, which ranged from 4.12 to 5.78, hit a minimum at k=2. This is an important additional metric to consider in model selection because minimizing the maximum error in the model helps prevent any one estimation produced by the model from being extremely far off the mark.

KNN Model Performance

We assessed the prediction accuracy of the KNN models by comparing the KNN estimates to point estimates using only the mean effort and SLOC. To remain consistent with LOOCV and prevent each test mission from influencing its own cost estimation, the mean was calculated as the numerical average of all other points in the historical dataset. For example, to estimate the effort of Dawn, we removed Dawn and averaged the effort of all other missions in the historical dataset, then calculated the MRE of that estimate using the actual effort for Dawn.

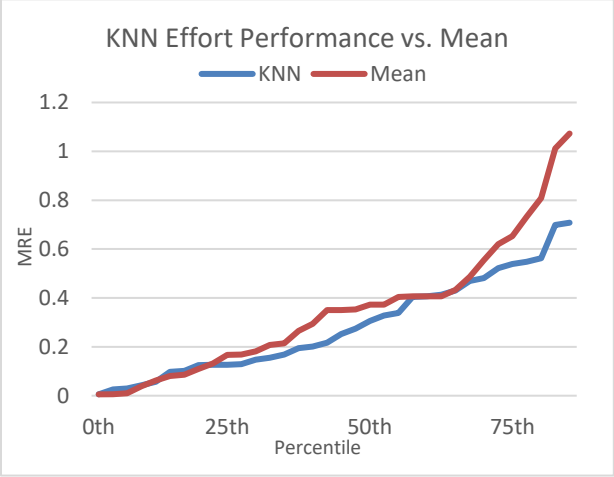

Figure 3 MRE distribution for KNN (k=4) versus MRE for predicting effort using the mean

Prediction performance for the KNN Effort model is considerably better than prediction performance using the mean effort, as seen in Figure 3. For this model, 50% of MRE values lie below 0.31, while the mean had a MdMRE of 0.37. The interquartile range of the MRE also was lower and narrower for KNN than for the mean; the maximum MRE was 39% lower. Visually, the MRE curve lies almost entirely below the MRE of the mean, indicating that most test missions had more accurate cost estimates when predicted using the KNN method.

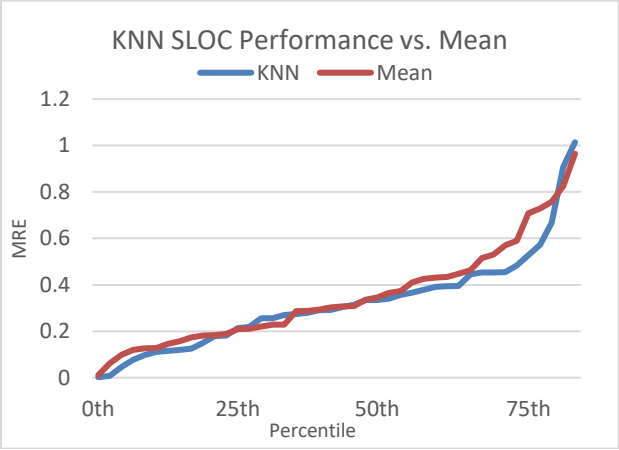

Figure 4 MRE distribution for KNN (k=4) versus MRE for predicting SLOC using the mean

For SLOC, the MRE curves lie almost on top of each other until the 50th percentile (Figure 4), indicating that the estimates had similar performances on the half of points that they predict "well". For the Upper 50th percentile, the KNN model predicted better than the mean and had a lower maximum MRE (not pictured).

Exploratory Analysis of Optional Inputs

Motivation—Traditional KNN prediction models require values for all input fields in order to output an accurate estimate of the response variable. This was true of the prior version of ASCoT KNN. However, since ASCoT's intended use is as an analogy-finder and cost-estimator during the pre-formulation phase of a project, having the option of flexible inputs is a feature that is potentially beneficial to a user who may be missing some inputs.

Since the previous version of ASCoT KNN had no method for dealing with missing inputs, it forced the user to guess the values of unknown parameters. Random guessing may impact which neighbors are utilized to build the cost estimates and thus which missions are returned to the user as analogous missions. This not only impacts the numerical output of the KNN models, but also the use of the tool as an analogy search engine.

In the new version of KNN with optional inputs, excluded fields are not considered in the calculation of neighbors and thus do not rule out any missions unnecessarily from the analogy search. However, while optional inputs returns more accurate analogies, the models built on reduced dimensions will likely perform worse than the full model with all attributes.

Evaluation of Number of Inputs—We ran a preliminary analysis on the KNN models to judge how much accuracy would be lost by using fewer inputs than the full set of seven possible parameters. Primarily using MRE, we compared using n=1, n=2, …, n=7 inputs to predict the effort level (for SLOC there are only six possible inputs so n=1 to n=6 were analyzed). For models with n < 7, LOOCV MREs were calculated for each mission as follows:

1. Set n
2. Select subset of n variables
3. Remove one mission from the data
4. Build KNN model on training missions using only subset of n variables and predict effort or SLOC for test mission
5. Compute MRE for that test mission
6. Repeat steps 3-5 for all missions
7. Return to step 2 with new subset of n variables until all subsets have MRE values
8. Average across all possible subsets, then return to step 1 with a new n

Thus, MRE values for each mission were calculated by computing all MREs for all subsets of n inputs, then averaging across all seven-choose-n models.

Computing the MRE values for each number of inputs resulted in an MRE distribution for n=1 to n=7 (n=6 for SLOC KNN Estimator). The same MRE and SE metrics as in the previous section were used to assess model performance.

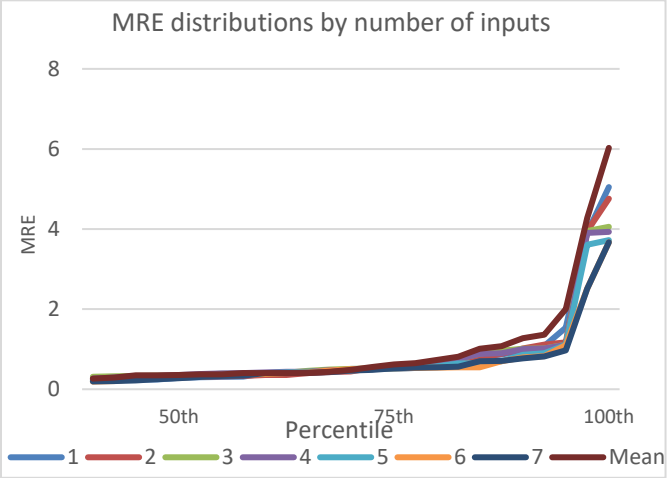

Figure 5 Upper 50th percentile of MRE distributions for n=1 to n=7 and mean

In Figure 5, observe that the MRE distributions are top of each other for the first half of the distribution, but perform in order of number of inputs for the second half. This effect is more easily seen in Table 18 below.

Table 18 KNN Effort errors by number of inputs

n	MRE					MSE	
	25th	50th	Mean	75th	Max	Median	Mean
1	0.16	0.30	0.59	0.59	5.05	45118	122698
2	0.20	0.31	0.60	0.60	4.76	61045	113427
3	0.21	0.36	0.61	0.61	4.06	64611	114799
4	0.20	0.38	0.59	0.59	3.93	56048	106790
5	0.20	0.32	0.54	0.54	3.72	44652	94804
6	0.14	0.32	0.48	0.48	3.67	27444	90549
7	0.13	0.31	0.47	0.47	3.67	37351	88276
Mean	0.17	0.37	0.69	0.69	6.03	40161	154793

As expected, the full model with all 7 variables predicted effort the best, with the lowest MRE and SE statistics across the board in Table 18. The 6 variable models were not far behind, still performing slightly better than the mean, especially in the 50th to 100th percentile and in mean MRE and MSE. Using only three to five variable inputs was more indistinguishable from predicting using the mean, except in the max MRE and MSE categories, indicating that the mean's prediction error distributions are highly skewed and the KNN error distributions are probably less so. Lastly, using only one or two inputs did not have much predictive power, but may still have value towards choosing analogous missions.

For the SLOC estimator detailed in Table 19, the full model with all six variables (no Inheritance) was the only model which performed better than the mean. Using five variables has about the same predictive power as using the mean, displaying comparable values for MRE, a higher MdSE statistic and a lower MSE statistic. However, it is important to note that since MRE values were averaged across all possible subsets, these tables do not capture information regarding how accurately individual subsets were able to predict the effort levels of each mission. The below average prediction error rates for n < 6 may be due to one or two subsets of n variables that performed especially poorly.

Table 19 KNN SLOC errors by number of inputs

n	MRE					MSE	
	25th	50th	Mean	75th	Max	Median	Mean
1	0.22	0.30	0.66	0.66	5.18	1633	5711
2	0.17	0.32	0.66	0.66	4.47	1755	5108
3	0.21	0.35	0.68	0.68	4.47	2000	4483
4	0.27	0.36	0.67	0.67	4.30	1722	4262
5	0.23	0.36	0.62	0.62	4.20	1605	3603
6	0.21	0.34	0.61	0.61	4.12	1159	3741
Mean	0.21	0.35	0.70	0.70	4.53	1375	6143

Evaluation of Attributes—Upon closer examination, different KNN models performed better than others depending on which variables were included. In order to compare performance between subsets, two MRE and squared error measures were computed separately for each subset using the same process as in the previous section (without averaging in Step 8). Then, each subset was given a point for each of the four error statistics that was lower than the mean statistic. Points were totaled across subsets and averaged across attributes to see which combinations of attributes performed the best.

In our analysis, we considered the performance of each attribute and the performance of each attribute pair. For effort, there were 7 different attributes, amounting to 64 subsets with each attribute individually and 32 subsets for each attribute pair. In our evaluation, assuming that the error statistics are equally weighted and independent, we would expect a score of 2 to indicate that a subset performed the same as the mean, a higher score to indicate superior performance and a lower score to inferior performance. However, it can be seen in Table 20 that all attributes score higher than 2. We attribute this to the correlation between error statistics.

Table 20 KNN Effort subset performance by attribute and attribute pair

	Size	No. Deploy	Redun	Type	No. Instr	Dest	Inher
Mission Size	2.66	2.97	2.84	3.00	2.38	2.75	2.91
Deployables	-	2.83	3.16	3.03	2.78	2.84	3.34
Redundancy	-	-	2.78	3.03	2.41	2.72	3.03
Mission Type	-	-	-	2.92	2.72	2.75	2.88
Instruments	-	-	-	-	2.16	2.13	2.44
Destination	-	-	-	-	-	2.45	2.66
Inheritance	-	-	-	-	-	-	2.72

For effort, the attribute which performed best on its own was Mission Type. Number of Instruments performed the worst on its own, marking error rates better than the mean on only 2.16 out of 4 error statistics on average. As a pair, the top two performers both included Number of Deployables, paired with Inheritance and Redundancy, respectively. Overall, the results suggest that including Mission Type or Number of Deployables is favorable to producing accurate estimates, while Number of Instruments is probably an ok measure to exclude, if necessary.

Table 21 KNN SLOC subset performance by attribute and attribute pair

	Size	No. Deploy	Redun.	Type	No. Instr.	Dest.
Mission Size	1.53	1.31	1.50	1.88	1.56	2.50
Deployables	-	1.25	1.25	1.63	1.25	2.13
Redundancy	-	-	1.44	1.75	1.19	2.00
Mission Type	-	-	-	2.00	1.63	2.44
Instruments	-	-	-	-	1.25	1.75
Destination	-	-	-	-	-	2.09

For the SLOC attributes, the results in Table 21 show that most attributes scored lower than 2, with the exception of Mission Type and Destination. Additionally, it is notable that while Number of Deployables was a good predictor of effort, it does not predict SLOC as well.

A similar analysis could be conducted for each of the combinations of 3 attributes, and so on. However, these results must be taken with a grain of salt because some of the error in the estimate will always be due to irreducible random error. Note that the irreducible error may be also be affecting which subsets perform above and below the estimates produced by the mean. Since the chosen performance metric involves a discrete binary component, random error has a large effect on our analysis.

Our intent in analyzing the performance of different subsets of inputs is not to say that users should only enter the inputs that performed well in the analysis. In reality, the user will not have much choice as to which variables are known or unknown. The purpose of our analysis is to inform users think critically about which inputs that they are entering in order to receive the best estimate possible and to give them an idea about which mission variables may be more important in deciding cost.

C. Cluster Analysis

Introduction—Cluster analysis, also called clustering, is an analytical approach for grouping a set of observations in such a way that members of the same group are more similar to each other than to observations in other groups [14]. This makes it a natural approach for developing analogy estimation models. There are a number of different statistical algorithms that can be used to perform clustering, four of which were used for cluster analysis in ASCoT as described below.

In cluster analysis, groupings are determined based on inherent differences within the data that cause a natural separation. Due to this separation between groupings in the data, the expectation is that the parameter of interest (the dependent variable) is different for each group and has smaller variance within each group than across the sample a whole. Unlike parameter estimation in regression, the dependent variable is not considered when defining the groups or clusters (performing cluster analysis) – this ensures unbiased groupings.[3]

An effective use of cluster analysis requires a blend of system knowledge and quantitative metrics. Much like linear regression, there are quantitative metrics that help decide whether an input variable or model is valid, or better than another model. However, also like linear regression, it is possible to 'over-fit' a model to a set of data. In the same way that linear regression can utilize a different number of input variables or different levels within a categorical variable, cluster analysis can utilize a different number of input variables and a different number of clusters. By examining multiple clustering methods, along with increasing levels of detail within those methods, we arrive at a precise yet robust set of clusters.

Because ASCoT cluster-based estimation works by assigning the user's mission to a group (cluster) of previous missions based on user inputs, and using the attributes of that cluster (KNN within the assigned cluster, average software effort months or SLOC within that cluster) to yield an estimation – having the right clusters is essential. Using the right clusters ensures robust estimates – even if some aspects of the input mission are not fully defined, if the input mission is assigned to the correct cluster, ASCoT will provide an accurate starting point for an estimate.

[3] In some cases of estimation through cluster analysis, the variance of the dependent variable is used as a minimization parameter, where clusters are chosen to minimize this spread per cluster.

In the following sections, each of the four cluster methods used are discussed along with analysis of the level of detail and the convergence of results within and across each cluster method. Additionally, the "Granularity Space", a framework for understanding the changing levels of detail across cluster methods is presented; and the balance between detailed, precise estimates versus robustness is discussed.

Cluster Analysis Overview

Cluster analysis is designed to identify different groups within a population, where differences between groups are maximized and differences within groups are minimized. Typically, cluster analysis methods rely on a Euclidean distance metric applied to normalized data to determine the distance between points and distances between points. This distance metric lives in the n-dimensional space of the data set, where n is the number of variables involved. In some cases, alternative distance metrics are used (e.g. Manhattan distance), or data sets are projected onto a principal component or kernel based space before clustering methods are applied.

For some clustering methods, one necessary input into the algorithm is the number of clusters that should be used. Typically, system knowledge informs an estimate of the number of distinct clusters expected in the data, and then quantitative metrics refine or fine-tune this number. Generally, these quantitative methods compare the distance of each observation to a hypothesized cluster center, assign each observation to the closest cluster center, re-calculate the cluster centers, and repeat the process until all points are assigned to a cluster and the cluster centers converge. This type of method may include additional nuance, including the use of principal components, random sampling, and multiple iterations of initial cluster centers to yield stable results.

Some cluster analysis methods do not require an expected number of clusters as an input– primarily hierarchical clustering and density based clustering. Hierarchical clustering breaks the entire data set down into a sorted tree, connecting each observation to its closest neighbor. In this methodology, major splits in the tree indicate different clusters. Density based clustering relies on a distance parameter P, where all points not within the specified distance P of each other create their own, new groups. In each of these cases, random starting points or iterations are not needed.

While there are many types of clustering that work well for different types and "shapes" of data, all of the methods rely on quantitative metrics to evaluate their performance and are subject to the level-of-detail versus robustness trade off, just like linear regression. The pros, cons, and applicability of each of the methods used are presented and discussed below.

Cluster Methods Used

In order to get full perspective of how historical missions group together, multiple clustering methods were compared across different levels of granularity, and the results were merged to form the final model used in ASCoT.

K-means Clustering (Centroid based iterative clustering)— k-means clustering is one of the most widely used clustering methods; it is performed on normalized data, requires a user input number of clusters ***n***, and requires iterations in order to arrive at a solution. K-means clustering works by choosing ***n*** random starting cluster centers, calculating the distance between each data point and each cluster center, then assigning each data point to the cluster center to which it is closest. Next, each cluster center is updated by averaging the components of every data point in that cluster, and the points are re-assigned to the (new) cluster to which it is closest. This process is repeated until the cluster centers stop changing and the members of each cluster are stationary. The number ***n*** of assumed cluster centers is a user input number. [14]

Once the cluster centers are stable for a given iteration of k-means clustering, the "sum-of-within-cluster-variance", a metric for the 'spread' or tightness of the clusters, is calculated. K-means clustering is then iterated many times, each time with different random initial cluster centers and the trial with the best sum-of-within-cluster-variance is selected as the final model.

The sum-of-within-cluster-variance is the sum of the distances from each data point to the center of the cluster to which it is assigned. This sum is recorded for each cluster, and summed over all of the clusters. Unlike an F-statistic for linear regression, there is no threshold which indicates an adequate sum-of-within-cluster-variance. However, the metric is still suitable for comparing models with different numbers of clusters and determining the best set of a clusters for a given ***n***. Due to the existence of local minima and sum-of-within-cluster-variance, it is necessary to iterate different initial cluster centers until the clusters yielding the best (lowest) sum-of-within-cluster-variance are found.

Choosing between numbers of clusters requires a balance of system knowledge, combined with analysis of the sum-of-within-cluster-variance. In theory, if every single point is assigned to its own cluster, the sum-of-within-cluster-variance would be zero; likewise, if the number of clusters is one, then the sum-of-within-cluster-variance would be the sum of differences from each point to the overall average of the data, per dimension. Neither of these results is desirable. The goal is to find a point where an increase in granularity, the detail which provides increased information (tighter groupings), is maximized but not over-defining the system by over-fitting the data. Usually, this is indicated by the point at which the step from ***n-1*** to ***n*** clusters yields a significant decrease in sum-of-within-cluster-variance, but a further step to ***n+1*** yields a very small decrease (Figure 6). This indicates a natural split of the data.

PCA Clustering—PCA Clustering works by first reducing the dimensionality of the data to the **k** principle components (k is based on choosing the principle components that account for the greatest spread in variance across the data), then performing k-means clustering on the data projected onto the **k** principle components [14].

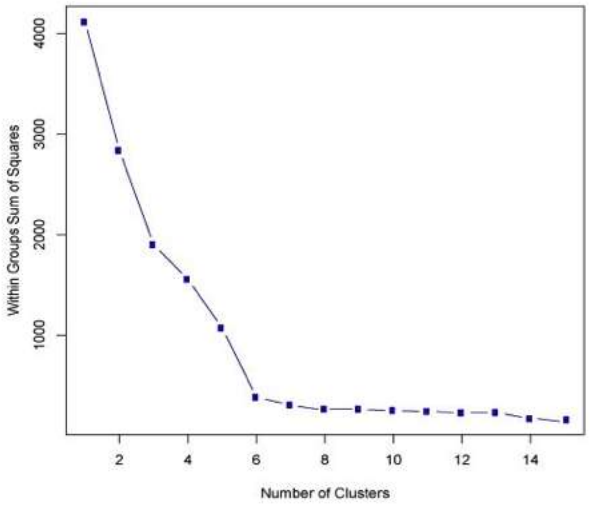

Figure 6 When choosing the number of clusters k, major kinks in the graph indicate significant decreases in sum-of-within-cluster-variance; here, it is highly likely that the optimal number of clusters is 6, but 3 should also be considered

By projecting the data onto **k** principle components, data that may be non-convex becomes more convex, which facilitates clustering. Additionally, by finding important sources of variance in the data, PCA Clustering automatically highlights the key dimensions of the data, serving to bring-forward the most differentiating factors.

Choosing the **k** used in PCA clustering adds yet another layer to model selection which affects the detail and precision vs. robustness tradeoff. The larger the value of **k**, the more dimensions of the data set are used in clustering, so this can increase precision – but using fewer principle components, key dimensions of the data, means that only the dimensions with the largest effects are used, which reduces noise and the potential for over-fitting.

Level of detail: number of principle components, number of clusters, iteration, result consistency, leave-one-out, variables and factor levels

Spectral Clustering—Spectral clustering is another method used in the present analysis which often builds off of centroid based clustering. In spectral clustering, a distance or similarity matrix is created, after which the graph Laplacian matrix of the similarity matrix is determined; with these steps, either centroid or density clustering may be performed[4]. Once we have the graph Laplacian, the major eigenvectors are taken, and the expected number of clusters is based on the number of major eigenvectors. Spectral clustering can work with non-convex data, but its specialty is more in the image and graph-edge-node based world.

Level of detail: m eigenvectors, type of difference matrix chosen (also present in other cluster methods, but more options are sometimes used here), leave-one-out, variables and levels

Hierarchical Clustering—Hierarchal clustering is performed in this analysis as another lens through which to look at our data, to find the optimal mission groupings, and to merge the results of other more nuanced cluster methods.

Hierarchical clustering has three main methods: (1) minimization of difference between closest points within a group, (2) minimization of distance between farthest apart members of each group, and 3) a mixture of (1) and (2), minimum distance to the center of a group. The intuitive methodology of hierarchical clustering is to repeatedly iterate over the entire data set, find the two points which are closest together, group those two points together or merge a point into the group of the point it's closest to, until all of the data has been grouped into a single tree-like structure [14]. In method (1), the linkages are determined as such: when a data point is closest to a data point that is already in a group, that new data point then joins that group, as in another branch of that group. In method (2) when comparing the distance of data points to points already in a group, a data point is compared the member which is farthest away in each group, and then it is connected to the group to which is has the smallest maximum distance. The third method compares a data point to the center of each current group of points, and assigns it to the group to which it is closest to the average.

Variables, Factors and Levels—When quantizing the levels of a categorical variable, the numeric values must be chosen carefully. If the different levels are not ordered (a categorical or nominal variable), then the information can be encoded in binary variables. However, if the information is ordinal, it must be encoded in numerical order. [2]

While using numeric variables exactly as they appear in the dataset, i.e. number of instruments, was a standard decision, decisions also needed to be made for the other, more qualitative, variables. For example, what is the best way to encode mission destination numerically? How can we best quantify orbiter missions versus rover missions?

[4] Note that density clustering adds another level of detail to be tuned in the granularity space. However, density based clustering is typically better for image and or character recognition – it is not ideal for globular data where the globules have different or unknown spreads or distances between points. As this is the case for the present data, density clustering is not used in this analysis.

Specifically, setting mission size as 1, 2, 3 and 4 makes sense – this is a typical encoding of size related categorical variables. Mission type is a bit less straight forward – how 'similar' are orbiters to landers? And orbiters to rovers? And landers to rovers? It is clear that landers and rovers are different than orbiters, and it makes sense that landers are closer to rovers than orbiters are to rovers – rovers include a lander and beyond that, have additional flight hardware.

Table 22 Choosing the best number of levels for categorical variables, as well as the right encoding values, is necessary to include the right information, while not over-segmenting the data

Variable	Levels	Encoding
Number of Instruments	Numeric, used as-is (6 instruments stays as 6, etc)	
Number of Deployables	Numeric, used as-is (2 deployables stays as 2, etc)	
Mission size	Small, Medium, Large, Flagship	1, 2, 3, 4
Mission Type	Orbiter, Observatory, Lander, Rover	1, 1, 2, 4
Destination	Earth, Inner Planetary, Asteroid/Comet, Outer Planetary	1, 2, 3, 4
Heritage	Very Low to None, Low, Medium, High, Very High	1, 2, 3, 4, 5
Redundancy	Single String, Dual String Cold, Dual String Warm	1, 2, 4

Since these assigned encoded variables will be used to differentiate and analogize between missions, we need the data to be representative of the qualitative variables that they represent. Therefore, the goal is to encode the qualitative information in an accurate, consistent, logical, and adequately representational manner.

Multiple cluster analysis iterations were performed, including or excluding different levels of categorical variable breakdown, and the results were generally consistent with the final results presented. The final variables (Table 22) chosen for use in cluster analysis, and the levels of categorical variables, were chosen for robustness, consistency in results, and alignment with system and engineering knowledge.

Level of detail: number of variables, levels of categorical variables including number of levels and quantity of encoding

Granularity Space: Choosing the right level of detail

So, there are many lenses through which to examine how missions group together, and each of these lenses has its own set of knobs – Now, how do we find the right combination of cluster analysis input parameters? How do we obtain the optimal amount of precision and detail while maintaining a robust set of clusters that does not over-fit the data?

Optimal results:

- Detailed groupings of missions that are similar to each other but different from other groupings
- Consistent across iterations
- Consistent between cluster methods
- Consistent to data variation (through LOOCV)
- Align with systems and engineering knowledge (if a Mars rover is grouped with a small Earth orbiter, something is off!)

In order to accomplish this, we need to

1. Do a parameter sweep from less detailed to more detailed, across the tunable input parameters
2. Iterate clustering to ensure we attain globally optimal clusters rather than getting stuck at local minima
3. Compare results across cluster methods to look for consistency and alignment at similar levels of detail
4. Map the results of which missions cluster together and where on the granularity space they align with each other, as well as systems and engineering knowledge

A facet of analysis that is tougher to quantize, relative to number of groups, number of principle components or number of eigenvectors, is the rate of convergence in cluster groupings. We can measure this rate by the number of samples required in order to 'find' a globally optimal result. In theory, while a large number of samples does not diminish the fact that the global minima was eventually found, it does provide commentary and context on the robustness of the clusters, and complexity and variance within the shape of the data.

Examining the convergence of cluster analysis methods at different levels of granularity ensures that we examine the optimal results of clustering per method, at each level within the granularity space in order to find the most robust yet precise cluster results (Figure 7).

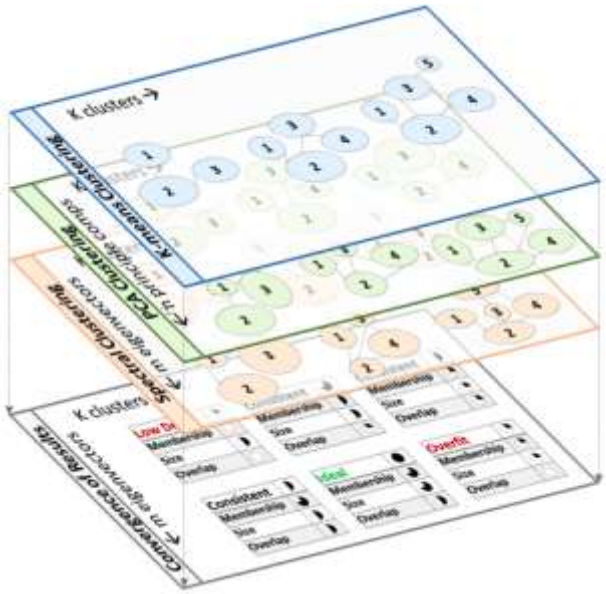

Figure 7 Looking at the results (Group Membership, Size of Clusters, Overlap between Clusters, and other parameters) across the Granularity Space, we can see where cluster methods and results align most, and at which level of detail; we're looking for consistency

The way we will view this is by examining which groups of missions 'break-off' from the rest of the missions, in what order. The order in which missions break off from the group of total missions to some extent indicates the degree to which they are 'different'. The order in which the mission types break off is discussed in more detail subsequently.

In order to tally the convergence of cluster groupings, and to determine how quickly distinct cluster groupings arose, additional analysis steps were implemented. Specifically, many iterations of clustering (across the whole granularity space) were performed, and a count of how often each mission was clustered with each other mission and with specific groups of missions was tracked. This was accomplished according to the following steps –

1. Choose a set of parameters (**n** clusters, on **k** principle components)
2. Perform clustering on the data
3. Record which missions appear in the same clusters using an assignment-comparison matrix (mission-pairing-comparison matrix)
4. Repeat for hundreds or thousands of iterations
5. Add the mission-pairing-comparison matrix to the prior mission-paring-matrix each time, yielding a tally of how often each mission was paired with each other mission
6. Perform hierarchical clustering on the assignment-comparison matrix in order to analyze and determine the groups of missions that were paired together most often

Furthermore, various iteration levels were employed within each cluster method— thousands of trials of k-means, PCA clustering and spectral clustering were performed with one iteration per trial, 25 iterations per trial, and 50 iterations per trial. This allows us to compare the consistency of local minima, how often the most common local minima occur compared to the global minimum[5] with different number of trial starts, and how long it may take the sum of local minima to converge to the global minima (if it does).

Cluster Analysis Results

To summarize, multiple cluster analysis methods were used, swept across the granularity space from less detailed to more detailed, the overall results from all of the methods were compared, and the most consistent clustering results were taken as the ultimate results. Inputs to the Ascot model assign a user's mission to a cluster from which a software cost or labor estimate is made based on KNN within the assigned cluster, and the average of that cluster is also shown. As our primary goal is correct and robust estimates that align with engineering knowledge, it is important that we create coherent, robust clusters, so that input missions get classified with the right analogous missions to create trustworthy estimates.

k-means clustering results—Because k-means clustering is less nuanced and more simple than PCA clustering, it is a good place to start to explore the data and the clusters that may be present. K-means clustering performed surprisingly well at a high level, breaking missions into relatively consistent groupings across iterations, and aligning decently with systems knowledge. It generally did well separating Rovers and Landers from other missions.

At the detailed, high granularity level, it also performed decently. Allowing up to 8 clusters, k-means created clusters that generally separated Earth missions, and rovers/landers, but had some strange mixes of large planetary orbiters and planetary and asteroid/comet missions. While some of this mixing makes sense, pushing towards 8 clusters with a data set that isn't large causes overfitting to become a cause for concern.

In the ranges between 3 and 8 clusters, k-means was a bit less consistent; it is this area, where we want to see which is a bigger differentiator – small versus large orbiter, planetary versus Earth mission, low versus high heritage, or Asteroid/Comet versus planetary – that we need more information.

[5] An interesting question emerged during our analysis: how does the most common local minima compare to the global minima? This would an interesting topic for a separate research endeavor. In this case, it was used to provide context and make inference about the shape of the data the more consistent the local minima groupings, and the closer they are to the global minima, the more globular the data and the more robust the cluster groupings, if they align.

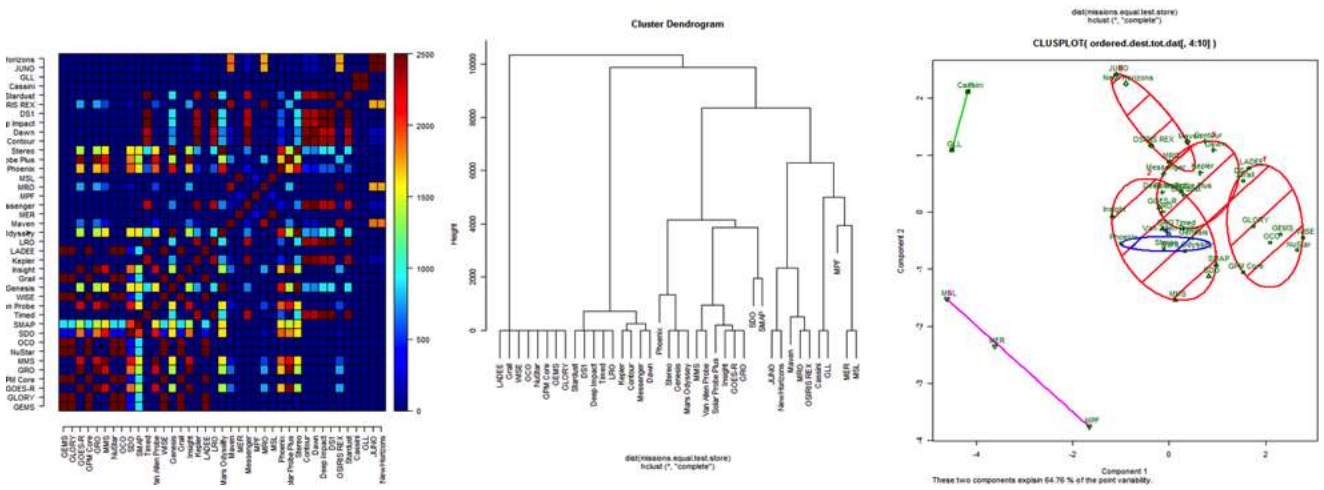

Figure 8 n=7 Clusters on k=3 Principle Components – the many non-red squares in the assignment-comparison matrix on the left, we can see that over many iterations, missions are not consistently grouped together; in the center plot, we can see less differentiation between groups in the hierarchical cluster analysis of the assignment-comparison matrix, based on the small height of some of the branches; the plot on the right shows mission groupings and group overlap, plotted with the first two principle components

PCA clustering results—Like with k-means clustering, PCA clustering was robust at the highest levels of separation. However, by tuning our level of detail, and finding the right place in the granularity space, PCA clustering was able to better sort some of the missions at the middle to high levels of detail. Specifically, PCA clustering with 6 principle components was able to provide our final definition of the set of Earth and small-to-mid size planetary orbiters.

Figure 9 shows the reduction in variance versus the number of principle components. At first glance, three looks like an appropriate number of principle components to use given the reduction in variance and the subsequent leveling-off. Six principle components looks like another viable option but since there are only seven original components, this removes principle components' attractive feature of dimensionality reduction.

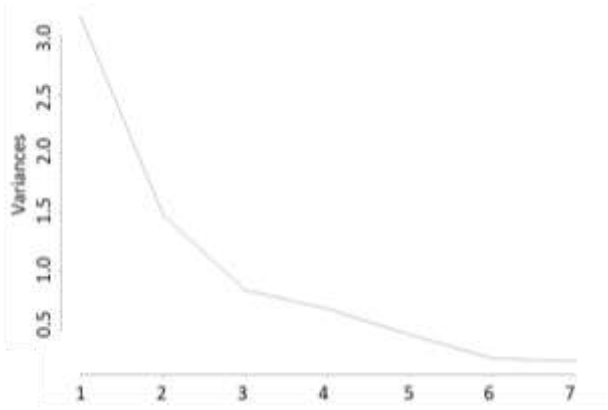

Figure 9 Remaining variance as a function of the number of principal components

As it turns out, using three principle components is stable and does well if we maintain a lower number of clusters (Large Outer Planetary, Rovers, Landers, and Earth & Planetary Non-Landers), but if we increase our level of detail enough to zoom in and separate out that Earth & Planetary Non-Landers group, increasing the number of principle components is beneficial. In some cases, at a lower detail level of the granularity space, with fewer principle components and clusters, the Rover missions group with the Large Outer Planetary missions. This is interesting, and makes sense given some of the characteristics of these missions: they are all large to flagship missions, have low heritage, many instruments, and mid to high levels of deployables. When increasing the level of granularity, they consistently separate – which also makes sense, and allows for better estimation for input missions.

Ultimately, seven Clusters based on six principle components were found: Smaller Earth Missions; Larger Earth with Planetary Missions; Smaller, Lower Heritage Planetary & Ast/Com Missions; Larger, Higher Heritage Planetary & Ast/Com Missions, Large Outer Planetary Missions; Rover Missions; and Lander Missions.

Spectral clustering results—Spectral clustering often does very well with image recognition and character recognition – and is a bit closer to a density based cluster analysis approach while still functioning on globular, non-image data. It may not be ideal, in this case, but it does provide a breakdown at a high level, and it is still helpful as part a lens when analyzing the problem.

Spectral clustering, like the other methods at the top level of granularity, did a good job of separating rover missions and flagship outer planetary missions. It is good to see this general consistency across different lenses for examining the separation of missions into groupings. However, within the small-to-mid-size mission groupings, spectral clustering did not do so well – it did not consistently group missions in this category into similar groups across iterations, and it didn't align well with systems and engineering knowledge.

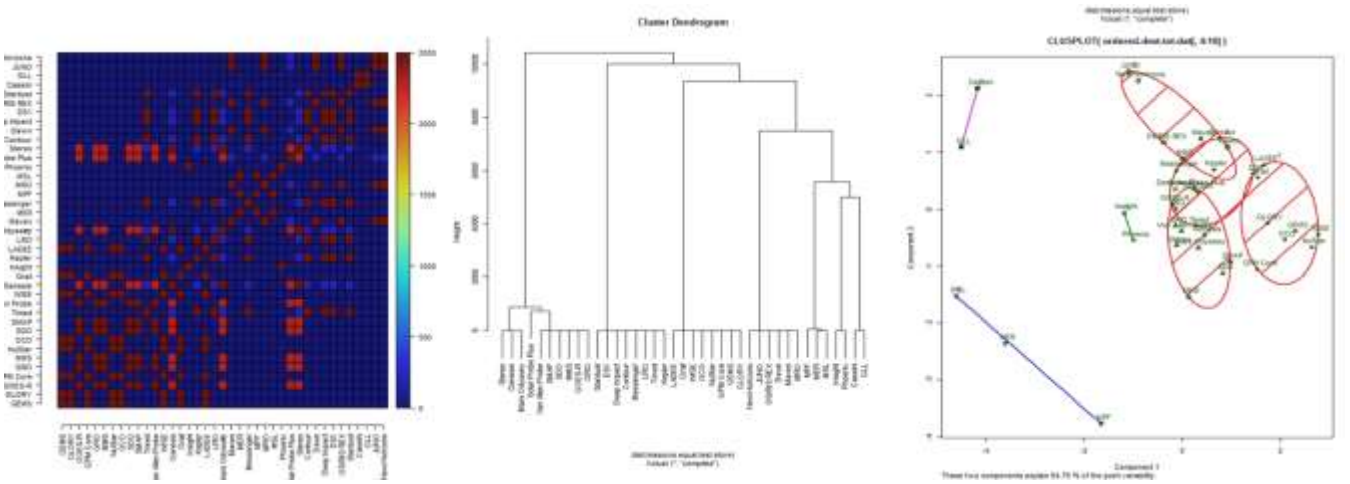

Figure 10 n=7 Clusters on k=6 Principle Components – the significant amount of dark red in the assignment-comparison matrix on the left shows that over many iterations, missions are almost always grouped together; in the center plot, we can see a great deal of differentiation between groups in the hierarchical cluster analysis of the assignment-comparison matrix, based on the large heights of the branches; the plot on the right shows mission groupings and group overlap, plotted with the first two principle components

Hierarchical clustering results—With Hierarchical clustering, we saw high-level breakouts between rover, planetary and earth based missions. This is not surprising given that the few Large Outer Planetary missions in the database are so different, and that there are very few Outer Planetary and Rovers missions. The different hierarchical clustering methods showed similar results. It's interesting to note – in some cases, the Rover missions were grouped with the Large Outer Planetary missions (as with some other methods) – this makes sense when we examine the characteristics of these missions, and see that they are all large to flagship size, and all have low heritage, many instruments, and mid to high levels of deployables. So, it's nice to see the quantitative clustering methods making sense, but we'd like to increase the granularity of analysis to align further with systems level knowledge – where Rovers form their own group.

Within the Earth and Planetary non-Lander missions (orbiters, observatories, flybys, anything that does not land softly on a planet), there was a good deal of confusion. This is not surprising, as this is a tough part of the data to differentiate, and hierarchical clustering is not the most nuanced option. Including or excluding some variables, or removing potential 'outlier' missions, yielded very similar high level results.

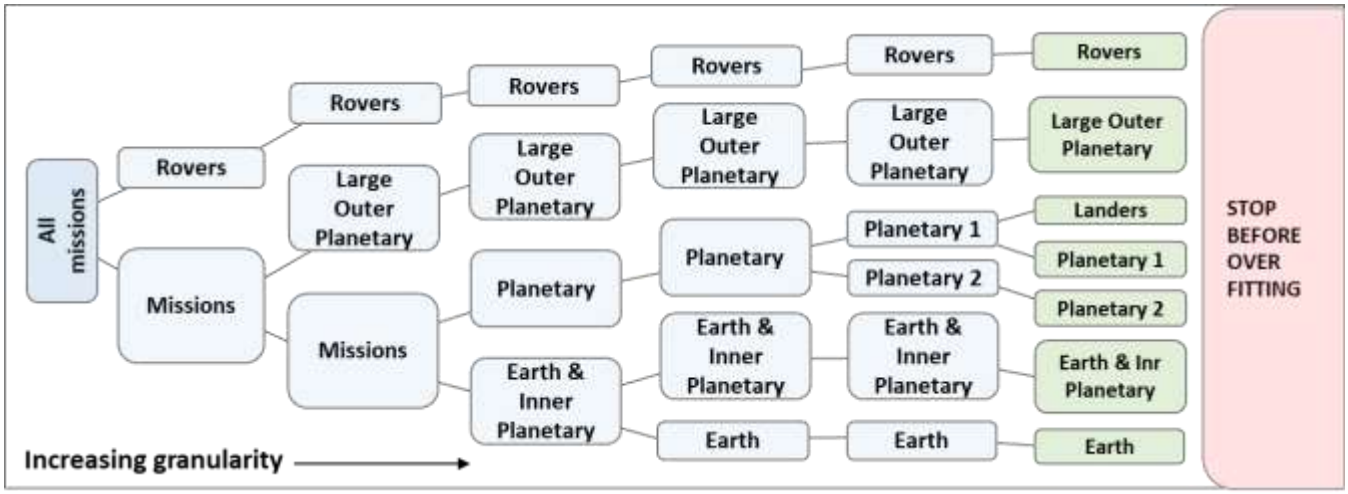

Figure 11 By examining which groups of missions split off in which order, we can get a sense for how 'different' those missions are from each other; note that the order in which mission groups split off is slightly different when comparing different clustering methods – but seeing similar final groups at the high-detail level of the Granularity Space increases our faith in the resulting clusters

Consistency Within Clusters/ Difference Between Clusters

When examining the convergence of results across the various methods considered, we see good alignment at the high level – either rover or flagship outer planetary missions typically break away from the data first into their own groups (Figure 10 and Figure 11), then Earth and Planetary missions separate. In some lower granularity tests, (fewer versus more clusters, principle components, variables and levels of categorical variables), flagship orbiters may group with rover missions. This makes sense, as the scale of rover missions and flagship outer planetary orbiters is often similar, and to this end, a complex examination of the characteristics per cluster shows that they do have overlap in size, low heritage, and high numbers of instruments and deployables.

We saw this result in multiple cluster methods at the lower end of the granularity space, but when using a more fine-toothed comb, we saw rover and flagship missions separate into different groups. This makes sense at a systems and engineering level, and while increasing the functionality of the resulting clusters in the context of a cluster based estimation tool, it decreases the variance in software cost and effort within those clusters.

One inconsistency that was observed across cluster methods was the time at which lander missions break off into their own cluster. While with a fine-tooth comb, we'd certainly expect landers to break away from orbiters, it makes sense that the breakoff could happen late in the process because a small lander mission may have complexity or cost that is similar to a complex planetary orbiter mission, especially when varying levels of heritage come into play. Even if different cluster methods separate them into their own group earlier or later as we sweep across the granularity space, the fact that we see them consistently form their own group, combined with system level knowledge, makes the results of our clustering analysis more credible.

The missions that were most hesitant to separate were the Earth and small to mid-size planetary and asteroid/comet non-Lander missions (including orbiters, observatories, and flybys/carriers – anything that doesn't land softly on a celestial body). This is not surprising since system and engineering knowledge shows that there is overlap between complex Earth orbiters, and especially observatories, and lower complexity planetary, and especially inner planetary, orbiter missions.

There was less consistency within the *order* in which different clustering methods separated out the Earth, Planetary, and Asteroid/Comet missions. Some cluster methods broke out Landers first, whereas others created multiple Planetary mission groups before separating the Earth missions, for example. This is not surprising, given system and engineering level knowledge. That different methods separated these missions into groups in different orders simply shows us that there is more variance in this part of our data space. Still, finding relatively consistent groupings at the higher end of the granularity space as well the more detailed level of our granularity space (more variance in the middle), makes us feel better about our results.

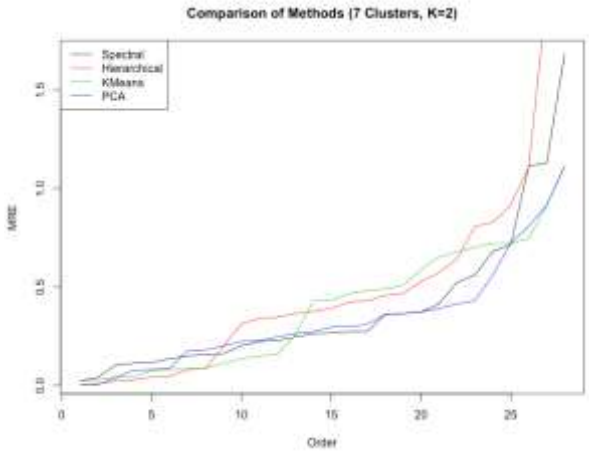

Figure 12 MRE Plots show that estimates from PCA Clustering perform well, and distinctly best, for the more extreme estimates

Overall, as we sweep across the parameter space, we see a convergence of results at the early stages: rovers and flagship outer planetary missions generally breaking into their own groups. We also see solid convergence at the higher end of the granularity space: a lander mission group, an Earth mission group, a smaller planetary and Earth mission group, and a final larger planetary and asteroid/comet group. Given the alignment of different cluster methods, the high level of iteration, sampling and convergence throughout the analysis process, and the alignment with system and engineering knowledge, we have more faith in the validity of the resulting clusters. Specific characteristics of each group are presented in the subsequent section.

It should be noted: in addition to finding clusters that are optimally separated mathematically, and are robust and repeatably separable, keeping the final use in mind (as an estimation tool) is part of finding the right clustering methodology.

For example, if a user inputs a large orbiter mission and sees their mission in a rover mission group, with a rover mission level software cost (whether or not this is true), a user may not trust the results, even though some flagship orbiter missions have a similar software cost to various rover missions. So, in the case of mathematically indistinguishable groupings, considering user experience and tool functionality in combination with system and engineering knowledge would be the most logical tiebreaker.

Additional testing for the quality of cluster separation of clusters was performed, with an emphasis on final applicability to software cost and effort prediction. As discussed elsewhere, prediction is made using the k-nearest-neighbor method in order to maximize estimation precision. The MRE of multiple clustering results was tested and is shown in Figure 12, where the minimum integral of the total

MRE curve is generally the optimal prediction result. PCA clustering generally performed best, especially with regards to less common missions (that create huge errors for some methods), where it offered a major reduction in total error.

Mission Characteristics by Cluster

By examining the missions per cluster, and the distributions and relationships of the variables per cluster, we can see the major separating factors and observe the similarities within groups. Each cluster's missions and their respective attributes are broken out in Figure 13 to Figure 19.

For Earth Missions, Number of Deployables, Mission Size, and Number of Instruments are the primary differentiators; for Planetary & Asteroid/Comet Missions: Heritage, Number of Instruments, and a summation of Mission Size with Destination differentiate the two primary non-Earth groups; Large Outer Planetary missions group together very well; Landers group together very well, and Rovers group together very well. The key differences between Earth and Planetary groups appear in the Heritage to Mission Size relationships, as well as the Number of Deployables; Landers and Rovers differentiate very clearly, based on the Number of Instruments and the Deployables, as well as Heritage.

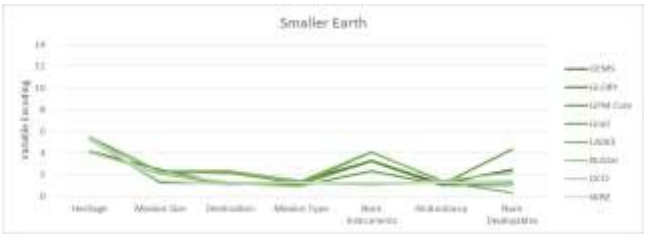

Figure 13 Smaller Earth Missions

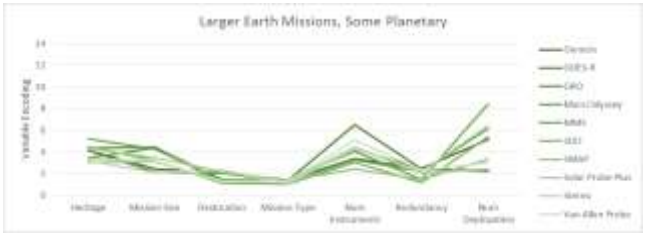

Figure 14 Larger Earth & Some Planetary Missions

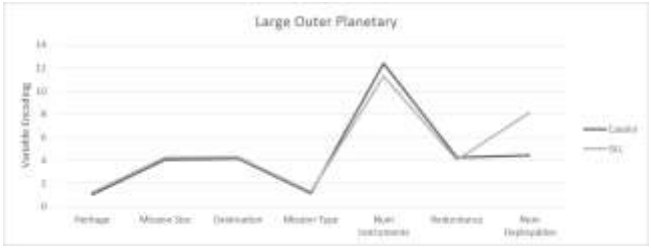

Figure 15 Large Outer Planetary Missions

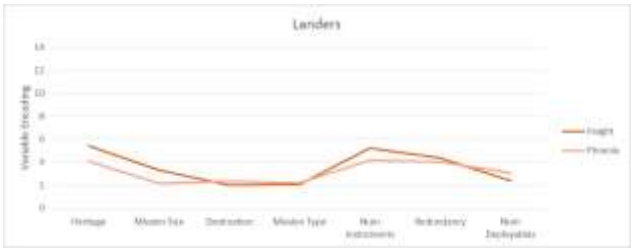

Figure 16 Lander Missions

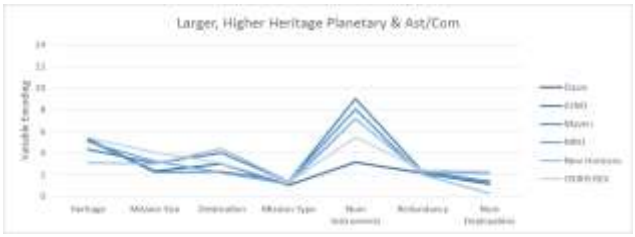

Figure 17 Larger, Higher Heritage Planetary & Asteroid/Comet Missions

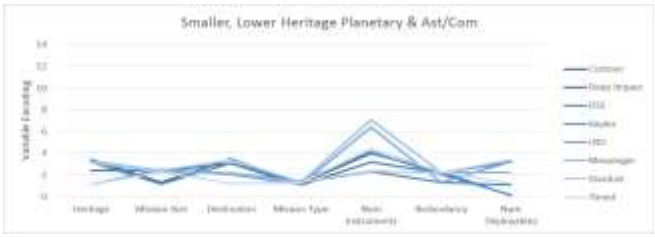

Figure 18 Smaller, Lower Heritage Planetary & Asteroid/Comet Missions

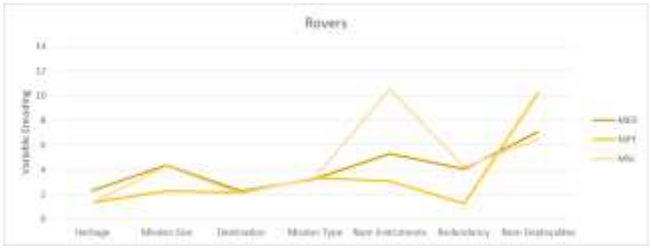

Figure 19 Rover Missions

Observing these similarities between groups increases our system and engineering level faith in the results, and may allow for additional engineering and mission cost and mass estimation profiles per group, to be examined outside of the present analysis.

5. CONCLUSIONS AND NEXT STEPS

The primary objectives of ASCoT are to 1) provide improved methods for estimating software cost and size in the early project life-cycle, 2) formalize analogy-based estimation taking advantage of machine learning methods, and 3) set a standard for the release of NASA wide estimation tools as on-line, web-based tools.

All of these objectives have been met. The fulfillment of the first two objectives are extensively documented in this paper. In meeting the third objective, a new task COMPACT (CubeSat Or Microsat Probabilistic + Analogies Cost Tool) has been started which will use the ASCoT framework.

ASCoT R1, NASA's first online web model was deployed to entire NASA audience with access to NASA ONCE in August 2018. The primary next step is the development of training material and delivering training across the various NASA centers. The training process will allow the team to collect feedback and test the process for pushing model updates remotely for updated observations or recalculated clusters.

Finally, the delivery of ASCoT R2, which will incorporate new mission data as it becomes available and updates as requested by our users, is planned for August 2019.

ACKNOWLEDGEMENT

© 2018. All rights reserved. The research was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.

REFERENCES

[1] Hihn, J.M. and H. Habib-agahi. Cost Estimation of Software Intensive Projects: A Survey of Current Practices. *Proceedings of the Thirteenth IEEE International Conference on Software Engineering*, May 13-16, 1991.

[2] L. Briand, K. El Emam, D. Surmann, & I. Wieczorek. An assessment and comparison of common software cost estimation modeling techniques. *21st International Conference on Software Engineering*, Los Angeles, CA, 1999.

[3] M. Shepperd & C. Schofield. Estimating Software Project Effort Using Analogies. *IEEE Transactions on Software Engineering*, 23(12), 1997.

[4] J. Hihn, T. Menzies, L. Juster, G. Mathew, J. Johnson, Improving and Expanding NASA Software Cost Estimation Methods, *2016 IEEE Aerospace Conference*, Big Sky, Mt., March, 2016.

[5] J. Hihn, M. Saing, E. Huntington, J. Johnson, T. Menzies, G. Mathew, J. Johnson,The NASA Analogy Software Cost Model: A Web-Based Cost Analysis Tool, *2017 IEEE Aerospace Conference*, Big Sky, Mt., March, 2017.

[6] Tim Menzies, D. Port, Z. Chen, J. Hihn, and S. Stukes. Validation methods for calibrating software effort models. In Proceedings, ICSE, 2005. Available from http://menzies.us/pdf/04coconut.pdf.

[7] Menzies, T. Chen Z, Port, D., Hihn, J., Simple Software Cost Analysis: safe or Unsafe?, *ACM SIGSOFT Software Engineering Notes* (SIGSOFT) 30(4)1-6, s2005.

[8] Tim Menzies, Zhihao Chen, Jairus Hihn, and Karen Lum. Selecting best practices for effort estimation. *IEEE Transactions on Software Engineering*, November 2006. Available from http://menzies.us/pdf/06coseekmo.pdf.

[9] "Stable Rankings for Different Effort Models" by Tim Menzies and Omid Jalali and Jairus Hihn and Dan Baker and Karen Lum. *Automated Software Engineering*, December 2010 .

[10] T. Menzies, · Y. Yang, · G. Mathew, ·B. Boehm ,· J. Hihn, Negative results for software effort estimation, *Empiracle Software Engineering*, October 2017, Volume 22, Issue 5.

[11] B.Boehm, *Software Engineering Economics*, Prentice Hall, 1981.

[12] B. Boehm, et. Al., Software Cost Estimation with COCOMO II, Prentice Hall, 2000.

[13] E. Kocaguneli, T. Menzies, and J.W. Keung. On the value of ensemble effort estimation. *IEEE Transactions on Software Engineering*, 38(6):1403–1416, Nov 2012.

[14] G. James, D. Witten, T. Hastie, & R. Tibshirani. *An Introduction to Statistical Learning with Applications in R*. New York: Springer, 2017.

BIOGRAPHY

***Jairus Hihn** (PhD University of Maryland) is a principal member of the engineering staff at the Jet Propulsion Laboratory and the manager of the Systems Modeling and Analysis Group. and is currently leading a laboratory wide cost improvement task. He has been developing estimation models and providing software and mission level cost estimation support to JPL's and NASA since 1988.*

Michael Saing *(BS, Cal State University, Long Beach). He completed his Aerospace Engineering undergraduate studies and gained his early career work experience at the NASA Ames Research Center. He is currently a Systems Engineer at the Jet Propulsion Laboratory developing aerospace engineering analysis models and serves as TeamXc's subsystems chair supporting NASA and JPL's spaceflight projects and programs.*

Alex Lumnah (*B.S., Occidental College, 2016) is a Systems Engineer at the Jet Propulsion Laboratory developing aerospace engineering analysis models and supporting Mars 2020 requirements development and V&V.*

Elinor Huntington *is a graduate student at Cal Poly Pomona, studying Computer Science. She works part time at JPL in the Office of Formulation. In a past academic life, she studied Russian Literature.*

Thomas Youmans (*MS Georgetown University) is a Systems Engineer and Data Scientist at NASA's Jet Propulsion Laboratory. Thomas received his Master of Science in Mathematics & Statistics from, and his Bachelor of Arts in Physics and Economics from Lewis & Clark College. Thomas supports early mission formulation through statistical modeling, operations and technical design through data analysis. Previously, Thomas served with the Peace Corps in Bulgaria.*

James Johnson *is responsible for providing Cost Estimates and Assessments, Schedule Estimates and Assessments, Risk Analyses, and Joint Cost Schedule Risk Analysis for the OCFO Strategic Investments Division (SID) at NASA Headquarters. His work for NASA HQ includes supporting high level Agency studies, providing support and consultation to projects, and developing policy and guidance for the Agency in the areas of cost, schedule, and risk assessments.*

Tim Menzies *(Ph.D., UNSW, 1995) is a full Professor in CS at North Carolina State University where he teaches software engineering and automated software engineering. His research relates to synergies between human and artificial intelligence, with particular application to data mining for software engineering.*

Melissa Hooke *is an undergraduate student at Pomona College, studying Mathematics with a concentration in Statistics. She currently works part time at JPL.*

APPENDIX A: ACRONYMS AND ABBREVIATIONS

A list of acronyms and abbreviations used in this paper, in alphabetical order, for reference.

Acronym	Definition
ASCoT	Analogy Software Costing Tool Suite
CADRe	Cost Analysis Data Requirement
CER	Cost Estimating Relationship
CML	Concept Maturity Level
COCOMO	Constructive Cost Model
JPL	Jet Propulsion Laboratory
KNN	K-Nearest Neighbors
LOOCV	Leave-One-Out Cross Validation
MdMRE	Median Magnitude of Relative Error
MdSE	Median Squared Error
MMRE	Mean Magnitude of Relative Error
MRE	Magnitude of Relative Error
MSE	Mean Squared Error
NASA	National Aeronautics and Space Administration
ONCE	One NASA Cost Engineering
PCA	Principle Components Analysis
SLOC	Source Lines of Code
WBS	Work Breakdown Structure

APPENDIX B: SYSTEM PARAMETERS WITH DEFINITIONS AND EXAMPLES

Detailed tables describing model inputs, complete with possible values, definitions, and examples for each of the seven inputs used in the KNN and clustering models.

Mission Type	**Values**	**Description**	**Example**
	Orbiter	A Robotic spacecraft that orbits or its target body. Also includes flyby spacecraft.	Aqua, New Horizons
	Observatory	Observatories are space based telescopes that support space based astronomy across a wide set of frequencies. They can be earth trailing or at the various LaGrange points created by the gravity fields of the earth, sun and moon.	Kepler
	Lander	A robotic spacecraft that does its science in-situ or from the surface of a solar system body. It does not move from its original location.	Phoenix
	Rover	A robotic spacecraft that does its science in-situ or from the surface of a solar system body and has the ability to move on the surface. To date all rovers have wheels but in the future they may crawl, walk or hop.	MSL
Destination	**Values**	**Description**	**Example**
	Earth	Missions that are in an Earth orbit.	OCO
	Inner Planetary	Missions that target planets within the asteroid belt. Also includes missions that are Heliocentric, Earth leading or trailing, at the Earth-Sun-Moon LaGrange points, and lunar mission.	Maven
	Asteroid/Comet	Missions that target asteroids or comets. As these may typically require more complex, or different, trajectories than inner planetary missions.	Dawn
	Outer Planetary	Outer Planetary missions are missions that travel beyond the asteroid belt.	JUNO
Number of Instruments	**Values**	**Description**	**Example**
	Number of Instruments	Total number of unique instruments on spacecraft.	Data ranges from 1 to 11 instruments. Median is 4 instruments.
Number of Deployables	**Values**	**Description**	**Example**
	Number of Deployables	Total number of unique deployables controlled by spacecraft.	Number of deployable Solar arrays, booms, robotic arms, etc. Data ranges from 0 to 10 deployments. Median is 3 deployables.

Flight Computer Redundancy	Values	Description	Example
	Single String	Spacecraft has no redundancy in the flight computer	Most Earth Orbiters
	Dual String - Cold backup	Spacecraft has redundant flight computers. Backup is normally off, is powered up and boots when prime string goes down	Most Deep space missions
	Dual String - Warm backup	Backup computer is powered on and monitoring state of prime computer, but does not need to maintain continuous operation (e.g., a sequence may be restarted, attitude control restarts with last known state, etc.)	MSL
Inheritance	**Values**	**Description**	**Example**
	Low to None	Total Inherited code, including modified code is < 10% of delivered code.	MER, TIMED, LRO
	Low	Total Inherited code, including modified code is between 10% to 20% of delivered code.	Deep Impact, New Horizons
	Medium	Total Inherited code, including modified code is >= 20% and < 50% of delivered code.	Messenger, MRO
	High	Total Inherited code, including modified code is >= 50% and < 80% of delivered code.	JUNO, SDO, GPM core
	Very High	Total Inherited code, including modified code is a minimum of 80% of delivered code.	MAVEN, Grail, NOAA-N-Prime
Total Mission Size	**Values**	**Description**	**Example**
	Small	Total Mission cost including operations in FY15 dollars is > $120M and < $220 million	Wise, small earth orbiters
	Medium	Total Mission cost including operations in FY15 dollars is > $220 million and < $600 million	Discovery class missions
	Large	Total Mission cost including operations in FY15 dollars is > $600 million and < $1.1 billion	New Frontiers class missions
	Very Large	Total Mission cost including operations in FY15 dollars is > $1.1 billion	Large assigned mission, MSL
Software Delivered Code	**Values**	**Description**	**Example**
	Small	Delivered logical lines of code is < 50 KSLOC	Small earth orbiters
	Medium	Delivered logical lines of code is > 50 KSLOC and < 120 KSLOC	LRO, Kepler
	Large	Delivered logical lines of code is > 120 KSLOC and < 220 KSLOC	LCROSS, SMAP, Phoenix
	Very Large	Delivered logical lines of code is > 220 KSLOC and < 300	OSIRIS-Rex, MER
	Extra Large	Delivered logical lines of code is > 300 KSLOC	MSL

APPENDIX C: MISSION DATA INCLUSION LIST

List of all missions included in ASCoT alphabetically and by which tool(s) they are included in.

Mission Name	Effort KNN	SLOC KNN	Clustering	CER
Aqua		✔		
Cassini		✔	✔	
Contour	✔	✔	✔	✔
Dawn	✔	✔	✔	✔
Deep Impact	✔	✔	✔	✔
Deep Space 1 (DS1)	✔	✔	✔	✔
Earth Observing 1 (EO1)		✔		✔
Fast Auroral Snapshot Explorer (FAST)				✔
Galileo (GLL)	✔	✔	✔	
Gamma Ray Observatory (GRO)	✔	✔	✔	
Genesis	✔	✔	✔	✔
GEOTAIL		✔		
GLAST		✔		
GLORY	✔	✔	✔	✔
GOES-R	✔	✔	✔	
GPM Core	✔	✔	✔	✔
Grail	✔	✔	✔	✔
Gravity and Extreme Magnetism Small Explorer (GEMS)	✔	✔	✔	✔
Hubble Space Telescope (HST)	✔	✔		
IBEX				✔
Insight	✔	✔	✔	✔
IRIS		✔		
JUNO	✔	✔	✔	✔
Kepler	✔	✔	✔	
LADEE	✔	✔	✔	✔
Landsat Data Continuity Mission (LDCM)		✔		
LCROSS		✔		
Lunar Reconnaissance Orbiter (LRO)	✔	✔	✔	✔
Magnetosqpheric Multiscale Mission (MMS)	✔	✔	✔	✔
MAP				✔
Mars Exploration Rover (MER)	✔	✔	✔	✔
Mars Odyssey	✔	✔	✔	✔
Mars Pathfinder (MPF)	✔	✔	✔	✔
Mars Reconnaissance Orbiter (MRO)	✔	✔	✔	✔
Mars Science Laboratory (MSL)	✔	✔	✔	✔
Maven	✔	✔	✔	✔
Messenger	✔	✔	✔	✔
Near	✔	✔		✔
New Horizons	✔	✔	✔	✔
NOAA-N-Prime		✔		
NuStar	✔	✔	✔	✔
Orbiting Carbon Observatory (OCO)	✔	✔	✔	✔
Orbiting Carbon Observatory 2 (OCO 2)				✔
OSIRIS REX	✔	✔	✔	✔

Phoenix	✔	✔	✔	✔
RHESSI		✔		
SAMPEX				✔
SMAP	✔	✔	✔	
Solar Dynamics Observatory (SDO)	✔	✔	✔	✔
Solar Probe Plus	✔	✔	✔	✔
Stardust	✔	✔	✔	
Stereo	✔	✔	✔	✔
Suomi National Polar-Orbiting Partnership (NPP)		✔		
SWAS				✔
Timed	✔	✔	✔	✔
TRACE				✔
TRMM				✔
Van Allen Probe	✔	✔	✔	✔
WIRE				✔
WISE	✔	✔	✔	✔